"This is the planet where nachos rule."

Outline

• What is Nachos?
  – Capabilities, purpose, history
• How does it work?
• How do I get started?
What is Nachos?

• An instructional operating system
• Includes many facets of a real OS:
  – Threads
  – Interrupts
  – Virtual Memory
  – I/O driven by interrupts
• You can (and will) modify and extend it
What else is Nachos?

• Nachos also contains some hardware simulation.
  – MIPS processor
    • Can handle MIPS code in standard COFF, except for floating point instructions
    • You can (and will) write code in C, compile it to MIPS and run it on Nachos.
  – Console
  – Network interface
  – Timer
Why Nachos?

• What better way to learn how an OS works than by building one?
• Much easier and more reasonable to build a simulated one in Java
• Skeleton code allows us to work on, replace, or upgrade one piece at a time.
History of Nachos

• Originally created here at Berkeley in 1992 in C++
• By Wayne A. Christopher, Steven J. Procter, and Thomas E. Anderson
• Used at many universities
• Rewritten in Java by Daniel Hettena
  – Now simpler, easier to grade, type-safe, portable, and more students now know Java.
How are we using it?

- Two Nachos assignments - “Phases”
- Phase 1 - Threading
- Phase 2 - Multiprogramming
How does Nachos work?

• Entirely written in Java
• Broken into Java packages:
  – nachos.ag (autograder classes)
  – nachos.machine (most of the action)
  – nachos.network (Phase 4)
  – nachos.security (tracks privilege)
  – nachos.threads (Phase 1)
  – nachos.userprog (Phase 2)
  – nachos.vm (Phase 3)
Booting Nachos

- When you run Nachos, it starts in nachos.machine.Machine.main
- Machine.main initializes devices - interrupt controller, timer, MIPS processor, console, file system
- Passes control to the autograder.
- AutoGrader will create a kernel and start it (this starts the OS)
The Machine!

- nachos.machine.Machine
- Kicks off the system, and provides access to various hardware devices:
  - Machine.interrupt()
  - Machine.timer()
  - Machine.console()
  - Machine.networkLink()
Interrupt Controller

• Kicks off hardware interrupts
• nachos.machine.Interrupt class maintains an event queue, clock
• Clock ticks under two conditions:
  – One tick for executing a MIPS instruction
  – Ten ticks for re-enabling interrupts
• After any tick, Interrupt checks for pending interrupts, and runs them.
• Calls device event handler, not software interrupt handler
Interrupt Controller (cont.)

• Important methods, accessible to other hardware simulation devices:
  – schedule() takes a time, handler
  – tick() takes a boolean (1 or 10 ticks)
  – checkIfDue() invokes due interrupts
  – enable()
  – disable()

• All hardware devices depend on interrupts - they don’t get threads.
Timer

• nachos.machine.Timer
• Hardware device causes interrupts about every 500 ticks (not exact)
• Important methods:
  – getTime() tells many ticks so far
  – setInterruptHandler() tells the timer what to do when it goes off
• Provides preemption
Serial Console

• Java interface nachos.machine.SerialConsole
• Contains methods:
  – `readByte()` returns one byte (or -1) and waits to interrupt when it has more
  – `writeByte()` takes one byte and waits to interrupt when it’s ready for more
  – `setInterruptHandlers()` tells the console who to call when it receives data or finishes sending data
• Normally implemented by nachos.machine.StandardConsole, hooked up to stdin and stdout
Other Hardware Devices

• Disk
  – Didn’t make the jump to Java from C++, we don’t use it for our Nachos assignments

• Network Link
  – Similar to console, but packet based.
  – Used for Phase 4.
  – You should be able to figure it out by then.
The Kernel

• Abstract class nachos.machine.Kernel
• Important methods
  – initialize() initializes the kernel, duh!
  – selfTest() performs test (not used by ag)
  – run() runs any user code (none for 1st phase)
  – terminate() Game over. Never returns.
• Each Phase will have its own Kernel subclass

Oh, how I hated the kernel, with his wee beady eyes, and smug look on his face! “Oh, you’re gonna buy my chicken!”
Threading

- Happens in package nachos.threads
- All Nachos threads are instances of nachos.thread.KThread (or subclass)
- KThread has status
  - New, Ready, Running, Blocked, Finished
- Every KThread also has a nachos.machine.TCB
- Internally implemented by Java threads
Running threads

• Create a java.lang.Runnable(), make a Kthread, and call fork().

• Example:

class Sprinter implements Runnable {
    public void run() {
        // run real fast
    }
}

Sprinter s = new Sprinter();
new KThread(s).fork();
Scheduler

- Some subclass of `nachos.machine.Scheduler`
- Creates `ThreadQueue` objects which decide what thread to run next.
- Defaults to `RoundRobinScheduler`
- Specified in Nachos configuration file
Nachos Configuration

• nachos.conf file lets you specify many options
  – which classes to use for Kernel, Scheduler
  – whether to be able to run user progs
  – etc.

• Different one for each project.
How to get started

- Go to class web page
- Download and install nachos package
- Read the README, make sure you can make proj1 OK
- The first phase will be posted soon with detailed instructions for first Nachos assignment
Advice

• One step at a time. Get a little bit working. Then a little more. Then a little more, etc.

• Find a good tool, including a debugger, and use it. One choice - Eclipse.
For More Information

• README file in the installation has lots of good stuff
• See the Class Web Page for intros, background, and the code itself.
• Read the code! You can see exactly what is going on.